

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Airey et al.	Examiner:	Wang, Jin-Cheng
Serial No.:	09/614,363	Art Unit:	2628
Date Filed:	July 12, 2000	Conf. No.:	2211
Title:	DISPLAY SYSTEM HAVING FLOATING POINT RASTERIZATION AND FLOATING POINT FRAMEBUFFERING		

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF JOHN M. AIREY UNDER 37 CFR §1.132

Dear Sir:

In support of the accompanying response to the February 20, 2008 office action in the above-referenced application, I hereby declare as follows:

1. I am the first-named joint inventor of the subject US patent application (hereinafter "the Application"), which is assigned serial number 09/614,363 and was filed on July 12, 2000.
2. I am submitting this declaration to show that some details of my invention claimed in the Application are described in US patent number 6,567,083 (hereinafter "the Patent"). In other words, using the language of MPEP 715.01(c), the Patent is a publication of my own invention. To that end, this declaration includes facts showing that:
 - I jointly made the invention upon which the relevant disclosure of the Patent is based,
 - I was associated with Daniel Baum, the first named inventor of the Patent, and

- Mr. Baum derived his knowledge of the relevant subject matter from me.

I jointly made the invention upon which the relevant disclosure of the Patent is based

3. In 1996, I was employed at Silicon Graphics, Inc. ("SGI") in Mountain View, CA as a staff engineer. In the summer of 1996, I started work on building a machine that used floating point rasterization, including floating point scan conversion, and a floating point framebuffer. The internal code name for the machine at SGI was "*Bali*." The Application describes and claims various aspects of *Bali*.

4. The general architecture of *Bali* is described in the document entitled "Bali Offsite," which is dated November 12, 1996 and attached as Exhibit A. This document was discussed among SGI personnel at an off-site meeting on November 12, 1996. Among other things, this document shows:

- A reference to the "R Chip" on the Agenda page (immediately following the cover page), the Contents page, and page 22. The R Chip refers to the portion of the *Bali* device performing rasterization,
- A chart of software simulating *Bali* on page 21,
- A block diagram of the rasterizer (*i.e.*, the R Chip) and scan converter on page 24,
- a block diagram of a floating-point block on page 29, and
- floating-point multipliers and floating-point normalizers/adders on page 35.

5. A number of other documents show specific details and aspects of my invention. These documents were prepared either by one of the joint inventors of the Application or by

someone with information that originated from one of the joint inventors. Some of those documents are discussed paragraphs 6 to 13.

6. "Bali Floating Point Representation" – A document dated December 3, 1996 discussing a 16-bit floating point format and internal variations used inside the R chip pipeline, and submitted herewith as Exhibit B.

7. "Mapping RenderMan to OpenGL" – A document I authored, dated January 24, 1997, that includes software code to be used with *Bali* hardware to have floating point processing through the R Chip pipeline, and submitted herewith as Exhibit C

8. Document containing Code headed "head 1.1" - A document I authored, dated January 24, 1997, that describes rasterization of pixels, color processing using floating point numbers and framebuffers, and submitted herewith as Exhibit D.

9. "Extended Range and/or Precision" - A document dated February 15, 1997 and authored by my co-inventor, Mark Peercy, and me. This document describes calculations using floating point numbers and which format of floating point numbering to use, and submitted herewith as Exhibit E.

10. "Tiny Floating Point" - A document dated February 15, 1997 and authored by Mr. Peercy and me. This document describes disadvantages of fixed point arithmetic and further describes a tiny IEEE like floating point format, and submitted herewith as Exhibit F.

11. Document containing Code headed "head 1.5" - A document with versions ranging from dates from July 15, 1997 to July 28, 1997 and authored by John Paul Alex, who was an intern working for Mr. Peercy and me. This document describes calculations using

floating point numbers and which format of floating point number to use, and submitted herewith as Exhibit G.

12. Email from me to Mr. Baum dated August 14, 1997. In this e-mail, I mention the s10e5 pipeline and high-speed data transfer from the host to the frame buffer, the frame buffer to the texture memory, and texture memory to the frame buffer. This document is attached as Exhibit H.

13. "Invention Disclosure" - A document dated September 22, 1997 that is an invention disclosure submitted to SGI Legal Services, and submitted herewith as Exhibit I. The document describes a 16 bit floating point format for texture store and framebuffer. The document further states that the invention was conceived about a year prior to the submission of the Invention Disclosure. This document also notes that Mr. Baum is the department manager for visual systems. Mr. Baum is not listed as an inventor.

14. Mark Leather, who was a member of the SGI technical staff from 1989-1997 but not an inventor on the Application, further corroborates my invention of the subject matter.

15. During a deposition conducted on December 7, 2007 (see Exhibit J), when asked if it was his "understanding that [the idea of multipassing data through the frame buffer] involved the use of floating point formatted data," Mr. Leather responded: "Yes, that was my understanding."

16. Further, in response to the question "And did you understand that that was [Mark Peercy's] concept that he was working on at SGI?," Mr. Leather responded: "I believe it was him and John Airey."

17. During the deposition, in response to the question “But do you recall the use of a floating point frame buffer as it related to Bali?” Mr. Leather responded: “Yes.” Further, in response to the question “And that was Dr. Percy and Dr. Airey’s work?” Mr. Leather responded: “Yeah.”

18. During the deposition, Mr. Leather recollected that a second project at SGI had a floating point frame buffer, but when asked if the “other work was an extension of Airey and Percy’s work,” Mr. Leather replied: “Yeah.”

I was associated with Daniel Baum at the time of my joint invention

19. Mr. Baum was my supervisor as the Hardware Director for the *Bali* project. We thus both worked together at SGI at the same time—among other times, while I was developing the invention claimed in the Application.

Daniel Baum derived his knowledge of the relevant subject matter from me

20. In his capacity as hardware director of the *Bali* project, Mr. Baum was responsible for understanding the technology and, as such, he was a decision-maker on the direction, schedule, and features of the project. In addition, in this capacity, Mr. Baum was responsible for updating company executives. To fulfill these and other responsibilities, Mr. Baum needed to be familiar with the software simulation, the floating-point scan converter, the floating-point frame buffer, and their technical advantages/benefits.

21. My co-inventors and I therefore described various aspects of the invention and the simulation to Mr. Baum so he could make the appropriate decisions. For example, Mr. Baum

was present at the above noted *Bali* offsite meeting of November 12, 1996 (see Exhibit A), as well as other internal *Bali* meetings. In addition, in his capacity as Hardware Director of *Bali*, Mr. Baum had access to many of the documents discussed herein.

22. As noted above, on August 14, 1997, Mr. Baum sent me an e-mail asking about opportunities to differentiate SGL. I responded, as noted above, by discussing the s10e5 pipeline and floating-point frame buffer (see Exhibit H). I thus conveyed details of my invention to Mr. Baum in this e-mail.

23. Danny Loh, one of the joint inventors of the Application, worked on the software simulation that appears to be mentioned in the Patent. Specifically, during a deposition conducted on November 9, 2007, Mr. Loh stated "So the question you asked me before, did I work with John Airey on the floating point on Bali?.... Yes." Mr. Loh further stated that he "wrote sort of the simulation framework to evaluate floating point formats in the frame buffer." When asked if Mr. Loh was doing that work with John Airey, Mr. Loh replied "with John Airey and Mark Percy." That portion of Mr. Loh's transcript is in the prior noted Exhibit J.

24. In his capacity as Hardware Director, Mr. Baum should have known about the software simulation.

25. It is my understanding that before the filing date of the Patent, SGI did not have another floating-point project that was independently derived. Instead, as Mr. Leather stated under oath, if any such projects existed, they were derived from the principles of my joint invention.

26. I hereby declare that all statements made herein of my own knowledge are true, that all statements made herein on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.



John M. Arrey

Dated: May 1st 2008

02839/00115 860759.1

pault

EXHIBIT A

Bali Offsite

11/12/96

Silicon Graphics Inc. Company Confidential

*Arch Work
Hennessy*

8
HIGHLY CONFIDENTIAL

S0861513

Agenda

9:00	BREAKFAST
9:15	Introduction
9:16	G Chip
10:45	BREAK
10:55	N Chip
11:15	Documentation Environment
11:45	Design Flow
12:00	LUNCH
12:45	R Chip
2:15	Software
2:30	M Chip
2:50	BREAK - Driving Range
3:30	D Chip
4:20	System
4:50	Wrap-up
5:15	AJOURN

Contents

G Chip	1
GE Ucode	3
GE vector and scalar	6
PE	10
GRU	13
N Chip	14
Documentation Environment	17
Simulation Environment	21
R Chip	22
Scan Conversion	24
Texture	27
Texture Filter	29
Lighting	30
FRU	38
SRAM	44
Pixel Response	48
Pixel SW	52
Geometry Pipeline	56
D Chip	58
System Configs	62
M Chip	65

G

Big changes:

Simpler vector unit
Commodity scalar unit

Special needs:

100 core
1kx128k port 200 Mhz ram
XTALK Net Interface

Interfaces:

XTALK, Onet, SDRAM, EEPROM, JTAG, NIC

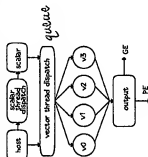
SGI Confidential

2

W

Vector Units Programming Model

- o HWP of four vector sub units (HWP was SIMP)
- o Host controls or scalar unit thread driven
- o Round Robin thread distribution (uninterrupted, v_0, v_1)



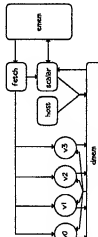
Vector Units Programming Model (cont'd)

- o Vector thread forks a single or a chain of vector code fragments - *inline code or unrolled loops*
- o Vector thread context switches of:
 - vector stream with pointer (end of file of scratch)
 - stream translation units, fragments
 - special stream access
- o Vector generates output to either the PE or the scalar unit (client)

for stitching vector code fragments

Vector Units Programming Model (cont'd)

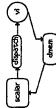
- o Unified memory:
 - internal memory - *data (HWP)*
 - shared and private per vector unit with a page per
 - shared word data path per vector unit
 - accessible by either host or scalar unit
 - read-write connected FIFO, immediate store and scratch
- external memory - *data (HWP)*
- shared and private per vector unit
- read-write connected FIFO, immediate store and scratch
- state backup



Equal word wide

Vector Units Programming Model (cont'd)

- o Synchronization - *since 2 ends of the (vector unit, loop) uses semaphore in stream*
- logical programmable threads
- scalar generated output resumes stream and file area
- dependencies
- host and scalar to dispatch
- vector to output
- vector to scalar



Transpose
a 4th region
of mem.



- Load/store RISC model (transposed load/store for surfaces and matrix operations)
- Direct and indirect (array of pointers)
- Persistent vertex state
- Vector lengths of 1 to 4 elements
- Single precision vector floating point operations (scaf, dot, cross)
- Generic conditional register move

Instruction set: i28b instr word

- four parallel control fields: load, store, floating point and sync
- local limitations of memory address, store base address + offset
- operand register
- arithmetic ops - all integer vector mask
 - three operands (self, 1st, 2nd)
 - 1st operand (16-bit register), 2nd, 3rd (16-bit register), cross
 - single operand post. inv, neg
- condition code:
 - per element conditional move
 - persistent condition code
 - condition code accumulation
- vector to vector spread

 $\text{scf-scale \& offset}$

- load an immediate address, scale offset operation, store a quad word to memory
- load quad word from memory, two element dot product, scalar to vector spread
- store quad word to memory, two element dot product, scalar to vector spread
- create product, conditional move of vector elements if = false

Geometric Statistics

- Run across seven representative applications/demos
- Most typical mesh size used: 4.
- Xform state changes are in about once per mesh in a non flattened CPU application
- Material calls reached couple of hundreds in a frame
1000 to per 400x lighting vs 125 pix.
- No use of target

Geometry Performance

- Best to perform tasks beyond the scope of transport layer
- Immediate (P0) vs list (PMA)
- Use mesh size of 4 for performance evaluation
- Vertex array as primary transport model
- State up for far geometric benchmarks default to normal acceleration and texture stream

Geometry Performance (cont'd)

- Initial figures based on vector mode cycle count (cpm - 55000000/sec, PMA - 8400000/sec)

Statistics:

- vertices in light no texture (L3/T)
- vertices in light no texture (L3/T, R/S, V/T)
- vertices in light no texture (L3/T, R/S, V/T)
- vertices local light, local viewer, no texture (L3/T, R/S, V/T)
- vertices local light, local viewer and texture (L3/T, R/S, V/T)
- solid mesh: 12 cubic, 2 vertices per vector unit, autonomous, vertex processing
- solid mesh: 12 cubic, 1 vertex per vector unit, autonomous, vertex processing

Type	CPU (mb) (M/sec)	DMA (mb) (M/sec)	TV (cycles)	40x4 (M/sec)	CPU (cycles)
vertex0	48.8	85.7	21	38.0	5
vertex1	51.1	87.5	21	38.0	5
vertex2	54.3	89.0	21	38.0	5
vertex3	57.1	91.5	21	38.0	5
solid mesh	na	na	200	8.0	2.5
solid coord	na	na	400	2.0	100

Geometry Performance (cont'd)

- One thread per vector sub unit
- Performance estimates meet our goal list...
- Risk - aggressive assumption on flush points latency
- Initial state:
 - vector to CPU ratio is between 77 - 1.05
 - vector to DMA ratio: 44 - 36

4x2 to speedup for vertex + evaluation respectively, over 4 kern.

Ball Offside II

Scalar Unit Requirements

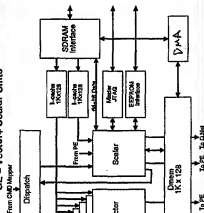
- ◆ 200 Mhz operation
- ◆ Floating point support for per-vertex lighting and clipping support. *+ vertex lighting for convolution.*
- ◆ G compiler, assembler, and debugger available.
- ◆ G-based simulator desirable.
- ◆ Verilog models available (behavioral and timing-accurate) for simulation.
- ◆ High bandwidth data path (64-bit) for Dmem interface.

November 11, 1964

1. **Introduction**

Ball Offsets II

GE - Vector+ Scalar Units



November 14, 1974

9

Ball Offsite H

Scalar Unit Options

Option 1:

Texas Instruments has a 200 Mhz DSP core that looks like it would meet our requirements.

Proj:

- ◆ The C7x performance numbers are:
1600 MIPS @ 200Mhz
800 MFLOPS single-precision @ 200Mhz
300 MFLOPS double-precision @ 200Mhz
- ◆ The C7x would be an on-chip solution with fast access to Dimen.

Con:

- ◆ TI plans on late Q2 '97 tape-out. Some risk if they slip.
- ◆ Unlikely option if TI is not our ASIC vendor.

November 11, 1968

Figure 3

Bali Offsite II

Scalar Unit Options

Option 1a:

A processor core provided by our ASIC vendor.

Pro

- ◆ Would provide a design flow integrated with the vendor.
- ◆ Would be an on-chip solution.

Car

- ◆ Current likely vendors (other than TI) do not seem to have a solution that meets the requirements.

November 11, 1924

1

SGI Confidential

7

Ball Offsite II

Scalar Unit Options

Option 2:

A core provided by a third party design house, internal MIPS core, or home-grown design. Includes using TI with a different ASIC vendor

Pros:

- ◆ Would be an on-chip solution.

Cons:

- ◆ Would have to find reliable third party design house that we could work out a design flow with.
- ◆ An internal MIPS port or home-grown design has increased complexity and risk.
- ◆ TI has not decided yet about licensing their core to another vendor

November 15, 1994

Page 6

Ball Offsite II

Scalar Unit Options

Option 3:

Off the shelf micro-processor external to the G chip.

Pros:

- ◆ Processors with 200Mhz operation and floating point are available.
- ◆ Off the shelf micro-processor probably has better L2 cache system than on-chip solution.
- ◆ Low risk for obtaining chips and development tools.

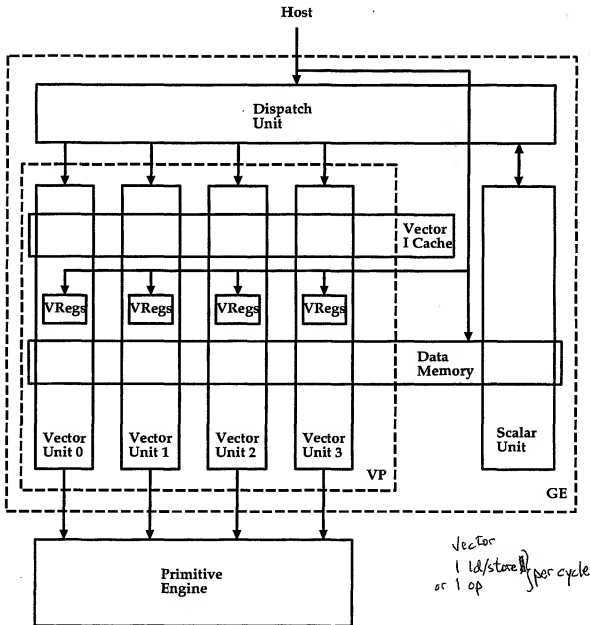
Cons:

- ◆ Data rate b/w from Dmem affected due to connection thru a 64-bit mixed address/data bus running at 100Mhz
- ◆ Increased pinout over internal solution. Increased cost?

November 15, 1994

Page 7

Vector Processor



VP	= 1M gates,	170K bits,	272 sq mm
VU	= 250K gates,	9K bits,	34 sq mm
DMem	= 5.8K gates,	384K bits,	24.4 sq mm
VIS	= 6.3K gates,	134K bits,	15.34 sq mm

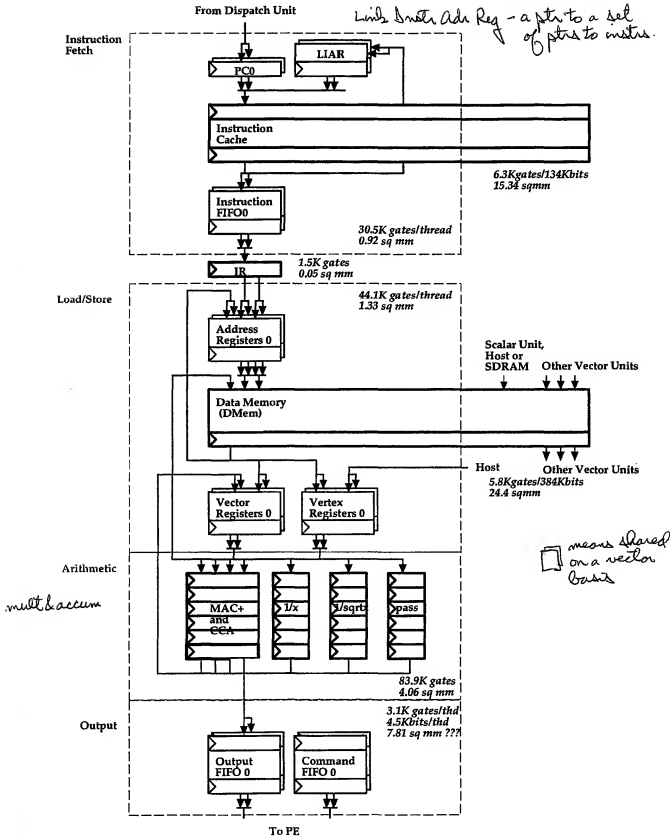
SGI Confidential

8

rog 11/9/96

Vector Unit Pipeline

250K gates for 2 threads
34.4 sq mm



SGI Confidential

rog 11/9/96

PE Design Notes

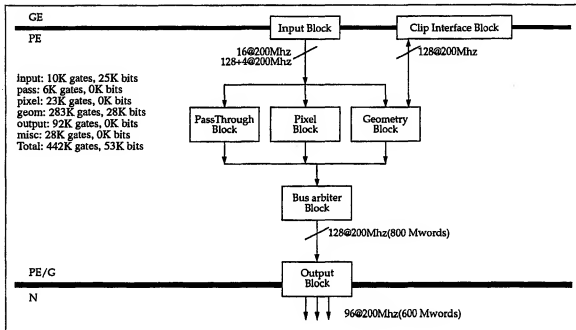
Rob (Skip) El-Kareh

Gloria (Globot) Lau

Erik (Bowzer) Lindholm

Paul (Ironman) Thilking

Mark (Flyboy) Young

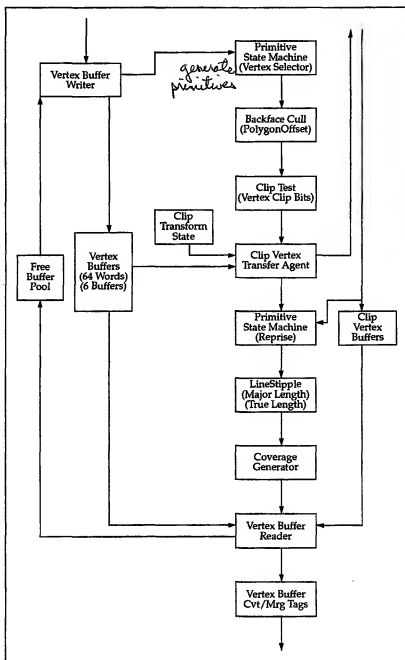


PE / Top Level Block Diagram

November 11, 1996 4:05 pm

Bali Offsite Notes

SGI Confidential 10



Geometry Block / Internal Block Diagram

PE Functional Goals

- GE Interface to Omega Network
 - Includes interface to Convert/Merge
- Pixel conversion, reformatting, and distribution
 - Input (IEEE Float, U16, U8), start aligned, no garbage
 - Output (S10E5, U16, U8), first pixel word aligned start of each packet
may happen in two in GE
- Bitmap unpacking, reformatting, and distribution, non-opaque and opaque, pixel packet per bit

November 11, 1994 4:00 pm

Bill O'Brien Notes

- Rasterizer State Management
 - Write / WriteThrough
 - Dirty Bits per State Set
- R Chip Synchronization
 - Broadcasts SyncID to R chips when Geometry is received before SyncFlag is cleared

Used for texture download

November 11, 1994 4:00 pm

Bill O'Brien Notes

- Geometry primitive processing and distribution
 - Standard OpenGL primitives (with swap)
trouble for this GL
 - PolygonMode, line and point generation
 - PolygonOffset, dz/dx & dz/dy, with scale and bias (may incur slight penalty)
 - LineStipple counter, major and true lengths, with first vertex of each segment
 - Clip exception processing (sent to GE, *WILL incur significant penalty*)
Waste extra info w/ vertices, so we can skip it back in this case.
 - Backface cull before clip
 - Zmin/Zmax/Hitflag for select
 - Bounding box coverage generation per primitive (including lines)
 - Feedback on Host or in GE
May be handled in GE or Host

November 11, 1994 4:00 pm

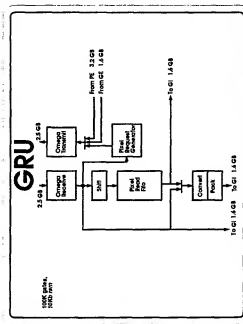
Bill O'Brien Notes

- Assumptions
 - R Handles: stippled line rasterization, wide stippled lines, wide points, AA lines and points
 - Bounding Box rasterizer selection is good enough for lines
- Open Issues
 - Who handles rasterization of Constant Multisample Points (Lightpoints)
 - What is the true benefit of R state shadowing? What is the state size? Can it be grouped into sets? Could it be better managed in N Chip?

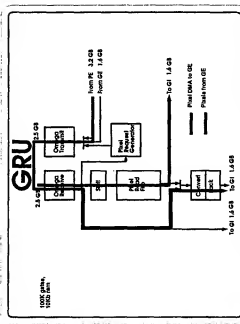
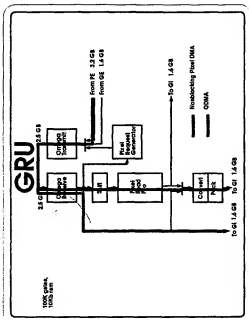
Material changes for per vertex lighting...?

November 11, 1994 4:00 pm

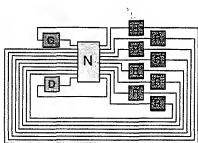
20 of 20



GE → lost goes out omega net
& back (saves pins)

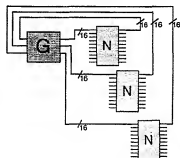


10-Node Configuration

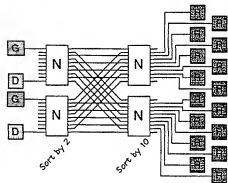


Was single 486 net.

Three 16-bit Channels Into and out of each Node

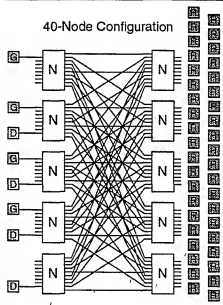


20-Node Configuration



12 N chips.

40-Node Configuration



32 R3

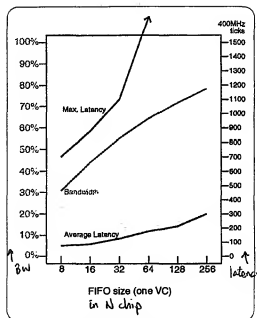
notice imbalance of ios.

Could use 4 chips, but no fair division.

SGI Confidential

N-Chip Issues

- * **Bandwidth and Latency**
We can get more bandwidth with deeper FIFOs.
We can get better latency with shallower FIFOs.
- * **Packet Ordering**
N-Chips can guarantee order within a channel, but there is no coordination between channels.
- * **System-level Deadlock**
Example: network clogged by texture requests so it cannot send texture responses.
One solution: virtual channels...one class can go when the others are blocked.
- * **Flow Control**
Throttle wire or credit scheme.
Will need separate one for each virtual channel.
- * **Error Handling**
At least want error detection.
Correction: ECC or retry protocol.



Packet Ordering

Only one path from Node A to Node B within Omega network, so packets on any one channel arrive in order.

But there is no ordering control (and wide latency variation) between 16-bit channels.

Most transfers are self-descriptive and do not care about ordering:

Texture requests
Texture responses
Video requests
Video responses
R = G Flow control

Others will have to cope using sequence numbers.

Because of latency variations, better to resequence after buffering...no waiting!

System-Level Deadlock

As network clogs, we need a way to relieve backpressure.

Must give precedence to "downstream" messages (e.g., responses instead of requests).

Virtual channels (sharing the same wires) is traditional way to allow high-precedence messages to flow despite snarl of low-precedence messages.

Virtual channels imply:

Separate buffering
Separate flow control
Intermingling between virtual channels (each transfer identifies its virtual channel)

We need to identify all possible deadlocks!

We need to categorize packets into virtual channels to assign precedence.

Example:

High	Medium	Low
Video request	Texture request	Triangles
Video response	R = G Throttle	Vertices
Texture response	Pixel read data	Read requests

SGI Confidential

Flow Control

Sometimes node cannot accept packets.
Example: R-chip can be overwhelmed by texture requests or pixel reads or triangles.

Need to be able to refuse packets.

Triangles handled by higher-level mechanism:
Throttle packets from R-G.

Others handled by flow control wires. + one per VC.

Two schemes:

Throttle: shut up ASAP.

Credit: keep track of receive buffers

Credit is more effective...hides the latency...but it presumes one set of receive buffers. If we have multiple FIFOs behind each input, credit scheme gets complicated.

Throttle scheme always works, but must set high-water-mark lower to skid due to latency.

Error Handling

Source-synchronous signalling can have errors unless it is tuned perfectly.

During bringup, "perfect tuning" can take months to achieve. Error correction can allow this effort to proceed in parallel with debug, not in series.

At least two possible schemes:
Error-correcting codes
Retry

Before we choose, we need to understand what kind of errors we expect.

ECC requires less hardware, but fixes fewer errors. It is well-suited to single bit errors, not bursts.

Retry requires a lot of hardware...big send buffers. More robust in case of many errors.

In both cases, handling errors at a finer grain requires less RAM...correct or retry micropackets..

N Chip Estimate

10 input ports, 10 output ports, 16-bit port

Die size:	14mm x 14mm
Total standard cell gates:	325.6K gates
Total ram bits:	125.0K bits
Total signal I/O:	~480 pins
Total package I/O:	~600 pins
Core frequency:	143MHz

Input unit (ISR, and decode, flow control)
Error handling (error detection and correction)
Performance monitor
diagnostic logic
Output unit (SSO, output arbitration)
Virtual channel file (100 x 40 x 16)
Virtual channel file (100 x 40 x 16)

SGI Confidential

Bail Documentation Methodology

Mark Leather

Steven Graphics - Confidential

How was Kona documented?

- Initial architecture documentation was done in showcase
- This was later turned into (official) transmitter spec documents for each ASIC
- Consistency of style was achieved through a published set of guidelines
- Templatelike sheets were made available to help enforce guidelines
- Externally generated documentation was imported by hand into transmitter
- Web site created using third party frame-to-html utility

Steven Graphics - Confidential

Kona register spec language

- Mark Goussman proposed a database language for describing registers
- Utilities were written to parse this language and create include files for C and Verilog
- Mark Leather later extended this to generate a web site containing drawings and descriptions of each register and field
- The register web site was very heavily used
- Incorporating this into transmitter documents created a problem. Register drawings needed to be converted to eps and imported into transmitter. Register and field descriptions needed to be re-written by hand.

Steven Graphics - Confidential

What did we learn?

- Formalizing guidelines was only loosely adhered to
- Transmitter documentation took up a considerable amount of the designers time
- Much of the documentation was done after chip layout, when it was least needed
- Showcase documentation was poorly organized and difficult to find
- Web connection mechanism sometimes did not work
- Importing the register stuff took so much work that it somewhat nullified the original objective.
- Everyone hates transmitter
- Whole documentation process consumed vast amounts of paper

Steven Graphics - Confidential

SGI Confidential

17

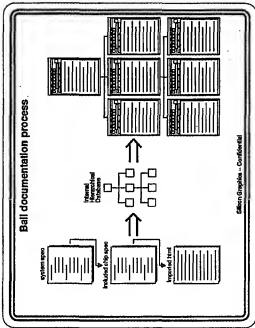
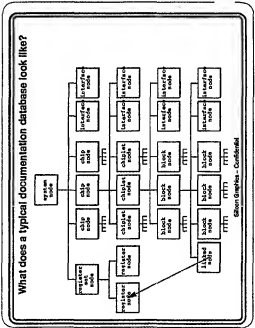
17

Ball documentation methodology

- Extended register definition languages to cover everything
- Ball documentation will consist of a hierarchical database containing node types for registers, interfaces, chips, chip blocks, algorithms etc.
- Database space format will be ASCII
- Multiple levels of include will be supported such that designer can work on smaller part of the database.
- A node finding facility will prevent duplication of work wherever possible, e.g. the description of "X Buffering" only needs to be entered once.
- Any fields requiring more than 8-31 lines of text or HTML can be expanded, allowing the use of existing HTML editors such as WebMagic, Gramma Creative or Navigator Gold.
- Databases used as input to a new tool which will create a complete web site

Edmund Granger - Confidential

- Extended register definition language to cover everything
- Full documentation will consist of a hierarchical database containing node types for registers, interconnects, outputs, chip blocks, algorithms etc.
- Database space format will be ASCII
- Multiple types of boards will be supported such that each designer can work on his/her part of the database.
- Node linking facility will prevent duplication of work wherever possible, e.g. the description of '2 Gallium' only needs to be entered once.
- Any fields requiring more than 2-3 lines of text or HTML can be imported, allowing the use of existing HTML editors such as Microsoft, Comma Create or Netscape.
- Database design is being used to create a complete web site

[illegible]

Form Number - Confidential

Node types currently supported	
node	This describes the complete system
processor	This describes a processor within the system
bus	This describes the bus within the system
chipset	This describes an ASIC within the system
chipset.etc.	This describes a smaller ASIC partition, usually related to a specific vendor module in the design hierarchy
lower faces	This describes a set of wires linking two or more blocks, chipsets or chips
features	This describes, in algorithmic terms, a specific block, a block, chipset, trip or system (from a 2D building).
package	This describes a specific package belonging to a given interface
package.etc.	This describes a specific package in the packed space
reg.leaf	This describes a set of registers belonging to the register set
reg.leaf.etc.	This describes a specific register in the register set
field	This describes a field of an interface, packet or register
field.etc.	This is a general node reserved for information which does not fit in any of the above categories. New node types may get added in order to better categorize this information as required.
misc	

Edwin Gayther - Corebentel

- | | |
|--|--|
| <p>This describes the complete system</p> <p>This describes a band within the system</p> <p>This describes an ASIC within the system</p> <p>This describes an ASIC partition entirely owned by one design team</p> <p>This describes a smaller ASIC partition, usually related to a specific wiring module in the design hierarchy</p> <p>This describes a set of wires linking two or more blocks, chips or chips</p> <p>This describes, in algorithmic terms, a specific feature of a block, chip, chip or system (such as a Z buffering).</p> <p>This describes a set of multi-core packages belonging to a given interface</p> <p>This describes a specific package in the package group</p> <p>This describes a set of registers implemented in the same address space</p> <p>This describes a set of registers implemented in different address spaces</p> <p>This describes a field in an instruction register</p> <p>This is a general notion reserved for information which does not fit in any of the above categories. New nodes may be added to order to better categorize this information as required.</p> | <p>This describes the complete system</p> <p>This describes a band within the system</p> <p>This describes an ASIC within the system</p> <p>This describes an ASIC partition entirely owned by one design team</p> <p>This describes a smaller ASIC partition, usually related to a specific wiring module in the design hierarchy</p> <p>This describes a set of wires linking two or more blocks, chips or chips</p> <p>This describes, in algorithmic terms, a specific feature of a block, chip, chip or system (such as a Z buffering).</p> <p>This describes a set of multi-core packages belonging to a given interface</p> <p>This describes a specific package in the package group</p> <p>This describes a set of registers implemented in the same address space</p> <p>This describes a set of registers implemented in different address spaces</p> <p>This describes a field in an instruction register</p> <p>This is a general notion reserved for information which does not fit in any of the above categories. New nodes may be added to order to better categorize this information as required.</p> |
|--|--|

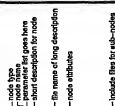
2008-2009

The diagram illustrates a database node structure with the following components and annotations:

- Database Node Structure:**
 - name:** A text field.
 - type:** A dropdown menu.
 - short_description:** A text field.
 - media_description:** A text field.
 - long_description:** A text field.
 - parameters:** A dropdown menu.
 - attributes:** A dropdown menu.
 - subparameters:** A dropdown menu.
 - subattributes:** A dropdown menu.
 - subsubparameters:** A dropdown menu.
 - subsubattributes:** A dropdown menu.
- Handwritten Annotations:**
 - html:** A bracket spanning the **media_description** and **long_description** fields.
 - like numbers one by one:** A bracket spanning the **parameters** and **attributes** dropdown menus.



© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 111–118

[illegible]

1

- How do we allow the spec language to evolve
 - o The language should be general enough to allow new node types and node attributes to be added or removed at any time during the project cycle
 - o It is very important not to break the document tree when new features are added
 - o Node attributes are assigned a priority: "optional", "recommended", "required" or "not recommended". A missing attribute generates a warning if "recommended", an error if "required". A specified attribute generates a warning if "not recommended", "not recommended".
 - o New required attributes are highly prioritized as "recommended"
 - o Attributes to be deleted are re-prioritized as "not recommended"
 - o Void one or more nodes until the documentation is warning free - then upgrade the new attribute to "required", or remove old attribute
 - o A web site will give a dated list of all spec language changes

Steven Springer - Contribute

- The language should be general enough to allow new rules types and node attributes to be added or removed at any time during the project cycle
- It is very important not to break the document tree when new features are added
- Node attributes are assigned a priority: "optional", "recommended", "required" or "not recommended", indicating attributes that are essential to a "well formed" or "not recommended". A specified attribute generates a warning if "not recommended". A specified attribute generates a warning if "not recommended".
- New required attributes are highly prioritized as "recommended"
- Attributes to be defined, but not prioritized as "not recommended"
- Mail one or more weeks until the documentation is writing free - then upgrade the new attributes to "required", or remove old attribute
- A week later will give a signal for the of type upgrade changes

© 2000 Pearson Education, Inc. All rights reserved. This publication is protected by copyright. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without permission in writing from Pearson Education, Inc.

- Printed output should be made possible, but actively discouraged
- Requires a book spec file – this sequentially lists all nodes to be printed
- Printed output will be in the form of a book, consisting of a list of contents, and a set of pages, containing a list of cross references at the bottom of each page.
- All hyperlinks will be converted into cross references. A cross reference consists of a page number if the reference exists in the current book, a list of book titles and page numbers, if the reference exists in other books, and a URL, for the on-line version of that reference.

- Printed output should be made possible, but actively discouraged
- Requires a book spec file – this sequentially lists all nodes to be printed
- Printed output will be in the form of a book, consisting of a list of contents, and a set of pages, containing a list of cross references at the bottom of each page.
- All hyperlinks will be converted into cross references. A cross reference consists of a page number if the reference exists in the current book, a list of book titles and page numbers, if the reference exists in other books, and a URL, if the online version of that reference.

10

Getting Started

- Official Ball Documentation Web Site
Look for this under the Ball Web Site in the next few days
- Language specification
Look for a fully hypertexted language spec in the Ball documentation site
Until this is available, a text based spec is available on the Ball tree under
gfs/MULT/tools/nrcs/lspec
- Example spec file
An example spec file is available on the Ball tree under
gfs/MULT/tools/nrcs/lspec
The resulting generated web site can be seen in
http://equila.rap
- HTML generation software
This can be found on the Ball tree under
gfs/MULT/tools/nrcs/
The usage is
spec <specfile> -s <destination_dir> <destination_url>
BallTemplate - Confirmed

SGI Confidential

20

Trial layouts, etc.

Bali Simulation

Define Design
Understand Risks

Put It Together
Sanity Check

Exhaustive Verification
Push Physical Design

Interface Review

Design Review

*integration of
chiplets then chips*

Design Review

95% Functionality
30% Correct

100% Functionality
90% Correct

Feature Set Defined

Set Design Rules

Set Coding Style

Write Chip Specs

Set Chip Interfaces

Set Chiplet Interfaces

Write High Level Sim (timing, layouts)

Write Chiplet Tests

Chiplet Testing

Chiplet Bug Fixes

*unit tests
(sanity checks
- nothing fancy)*

Set Verification Rules

Write Chip Test Specs

Write System Test Specs

Make Test Jig Creator Tool

Test Jig Creation

Write Chip Sanity Tests

Write System Sanity Tests

"IntensiveTest" writing Period (Asim and RTL)

Chip Verification In Asim and RTL

System Verification In Asim and RTL

Write Fastsim C Simulator

uocode and system sw development

Code Development using Fastsim and Asim

*Chip integration - System integration
in Asim*

Regressions on Asim

Regressions on RTL

Asim Bug Fixes

RTL Bug Fixes

Write Synthesizable RTL, Timing and Layout

*Verilog Asim has
advantage of recycling
of smv + avoid compile
overhead of C smv
to verilog RTL.*

SGI Confidential

21

*Asim = verilog C. Not defined. Transaction accurate,
maybe cycle acc.*

X Chip Team

Patrick Law
Chris Michel
Steve Nigam
Ray Nagel
David Wang
Alice Nish
Nobuo Tsunemitsu
Walt Grossman
Alex Nishin
David Tsunemitsu
Paul Melle
Mark Lechner
(M Chip)
Home Page: <http://ps.sgi.com/html/press/4000/chipm7/index.html>

Requirements

Area-optimized fast SHARX
Isarman
Real "Y" network simulations
On-the-fly texture compression
Need to simulate new fog, lighting
Multiple OS injection
External Paria
420 Mhz clock generator
140 Mhz SHARX SHARX, also INCI6 and 4000 SHARX
M chips (--)

Sign-off items.

New Functionality

Extended range & precision color components (alpha float)
Per-pixel lighting
Fog is a function of range & elevation
2 Active Textures
Space-varying convolve

Performance

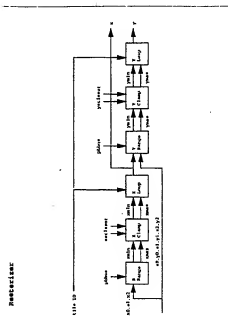
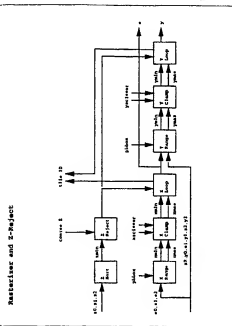
140 Mhz clock (SHARX vendor consensus)
1 pixel per clock polygon fill rate (2 tex, alpha-blend, & text)
1 pixel per clock drawpixels
1/2 pixel per clock accumulation
1/4 pixel per clock 7x7 convolve
4 pixel per clock clear
64 bits per clock texture download

(copy of Tex format)

SGI Confidential

HIGHLY CONFIDENTIAL





Eye-space Barycentric Coordinates

$$f = A * E0 + B * E1 + C * E2$$

* color, normal, eye-space depth

$$A = \frac{e1e2}{e1e2 - b1e2 + c1e2}$$

$$B = \frac{e2e0}{e2e0 - b2e0 + c2e0}$$

$$C = 1 - A - B$$

* texture coordinates

$$A = \frac{e1e2}{e1e2 - b1e2 + c1e2}$$

$$B = \frac{e2e0}{e2e0 - b2e0 + c2e0}$$

$$C = \frac{e0e1}{e0e1 - b0e1 + c0e1}$$

* screen-space depth

$$A = a$$

$$B = b$$

$$C = c$$

where

$$Atp, pt, p0$$

$$Atp, pt, p1$$

$$Atp, pt, p2$$

$$Atp, pt, p3$$

$$Atp, pt, p4$$

$$Atp, pt, p5$$

assumes q
sends $\frac{1}{w}, q/w$


250kg 1st beam cov.
350kg 2nd "

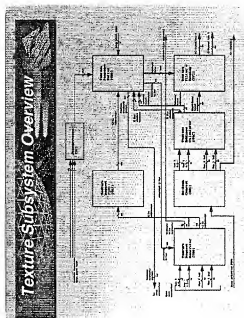
May need to go to
multi-pass beam
cov to save gates.

SGI Confidential

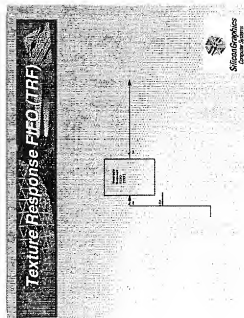
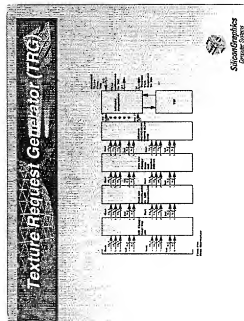
Texture Subsystem

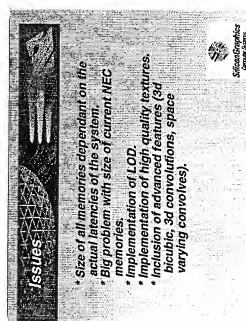
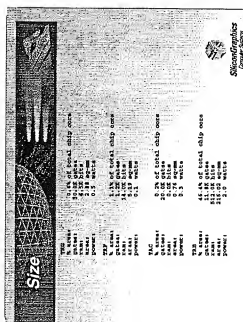
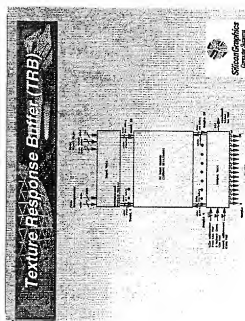
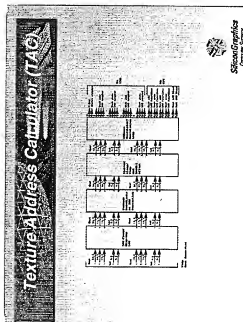
- * Overview
- * TRG
- * TRF
- * TAC
- * TRB
- * TFI (dignam)
- * Size
- * Issues

 Silicon Graphics
Imaging Division



27

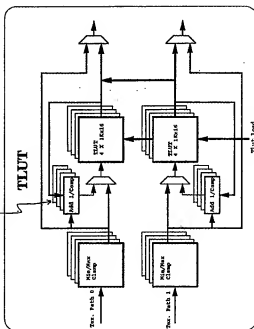
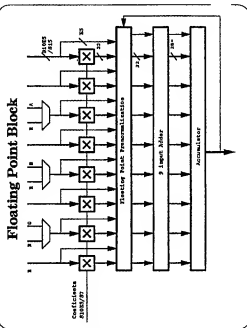
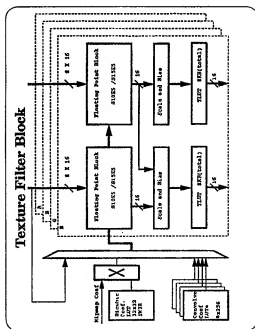




Texture Filter and Imaging Features

Supports:

1. All the old stuff
2. Blending of two simultaneous textures
3. 2x1 anisotropic filtering
4. High quality fully anisotropic filtering (aka video filtering)
5. Floating point support (for SLOs and S15)
6. BI-cubic filtering in one pass
7. 3D Bicubic for medical imaging
8. Coefficient table and support for float per component convolution
9. Min/Max and histogram support using TLUT & List merging
10. Color matrix conversions
11. Space variant convolutions



For histogram support - counts hits & does min/max op.

Started as a
16-bit true color
and to change
since need 32

R Lighting

R Lighting

Amy Wigdahl
 David Wang
 John Airey
 Nyl V. Sines
 Rick B. Smith
 Bob Drablin
 Erik Lindholm
 Mark Peercy
 David Tannenbaum

<http://hp.usd/real/rees/doc/chapter/lighting/lighting.html>

Ball Offsite 2

R Lighting

Functionality

- Lighting
- Texture Environment
- Fog
- Alpha and Mask Logic

Ball Offsite 2

R Lighting

Features

- full speed (143 Mpix/sec)
- per-pixel lighting (Slim-Phong)
- tangent space bump mapping
- environment (cube) mapping
- texture as material color or specular shininess
- two-sided lighting
- normal vector bypass
- distance attenuation
- spot light attenuation

Ball Offsite 2

R Lighting

Features, cont.

- two texture environments
- range based fog
- altitude base fog
- dithered alpha-to-mask conversion
- alpha-to-one
- alpha function

Ball Offsite 2

SGI Confidential

R Lighting

Per-Pixel Lighting

Implemented in Lighter:

$$Lc = Att * Spot * Cl ($$

$$+ Sm * (N * L)$$

$$+ Sm * (N * R) * n$$

$$)$$

$$SpotL = (S - L) * cos(CUT_OFF_ANGLE)$$

$$(S - L) * SPOT_EXP : 0$$

$$= 1 \text{ if disabled}$$

Implemented in TVW:

$$C = Sm + Att * As + S * Lc + Sm * X$$

Note: X means X can come from texture

-Ball Offset 2

R Lighting

Fog Calculation

$$range = \sqrt{X^2 + Y^2 + Z^2}$$

$$deltaAltitude = AltitudeEye - Altitude$$

$$f = exp(|FogEye - Fog| * range * 1/deltaAltitude)$$

Fog values are determined by indexing LUT at altitude of eye and altitude of fragment.

-Ball Offset 2

R Lighting

Texture Environment

$$CV = A * Cf + B * Ct + D$$

Implements lighting equation

Implements TVW functions, including AND

MapTex Fragment:	A = 0	B = 0	D = 0
Texture:	A = Ct	B = 0	D = 0
Modulate:	A = 0	B = 0	D = Cf
Blend:	A = -Ac	B = Ac	D = Cf
ADD:	A = -1	B = Cc	D = Cb

Ct: texture color
 Cb: texture blue
 Cg: texture green
 Cd: texture color
 Cc: constant color
 Cn: base color

-Ball Offset 2

R Lighting

Precision

Nxyz, Lxyz, Rxyz - S15

Colors - (s15s, s15s, s15s, s15s)

Att - 0.16

texture - (s15s, s15s, s15s, s15s)

texture as normal - clamp to S15

-Ball Offset 2

SGI Confidential

-R Lighting

Precision, cont.

mask - 8
 X,Y,Z (screen) - (12, 12, #31)
 dxdk, dxdy - #15
 exp - 5
 X,Y,Z (eye) - #23
 altitude (eye) - 12

-Ball Offsite 2

-R Lighting

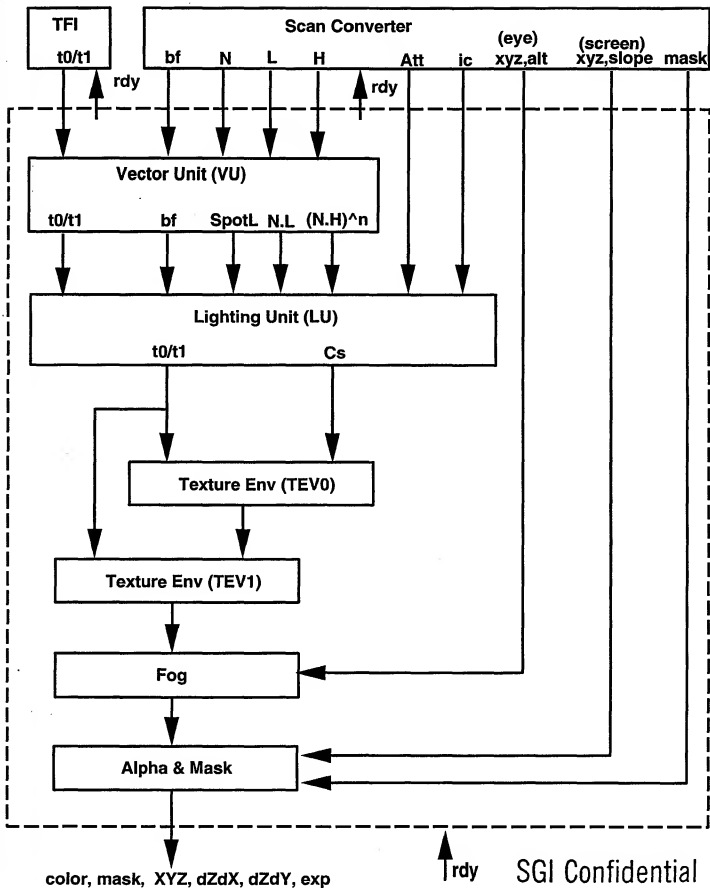
Issues

- Spot light aliasing - linear fall-off?
- pre-lighting xxy
- Interpolated Att OK?
- Approximate range for fog OK?
- Z-based fog also necessary?
- Can per-pixel lighting be used instead of dedicated fog hardware?

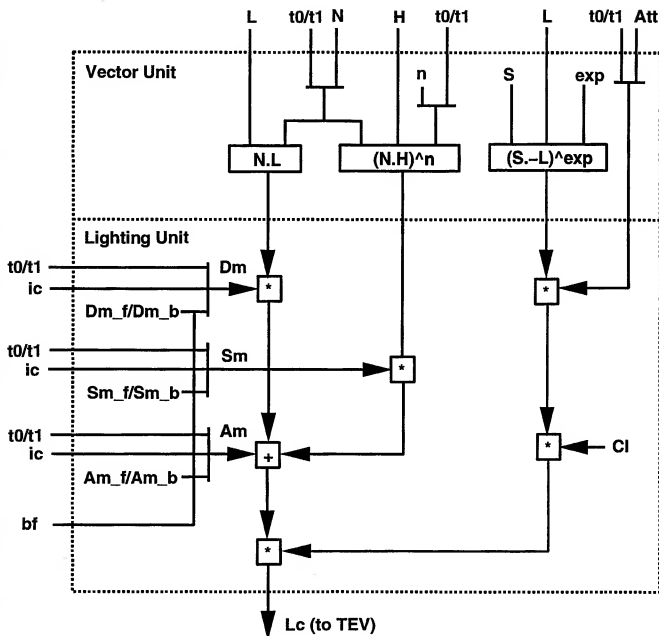
-Ball Offsite 2

SGL Confidential

LTF Block Diagram



Lighter Block Diagram



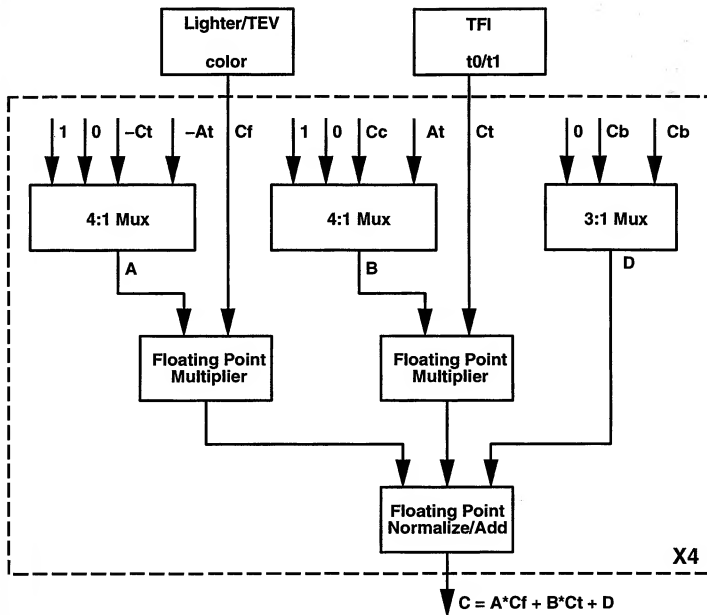
```

Lc = Att*SpotL*Cl { // distance*spot light attenuation*light color
    Am              // ambient material*ambient light
    + Dm*(N.L)      // diffuse material*diffuse light
    + Sm*(N.H)^n    // specular material*specular light
}
    
```

ic = interpolated color
 $t0/t1$ = textures
 $*_f$ = front material color
 $*_b$ = back material color

SGI Confidential

Texture Environment Block Diagram



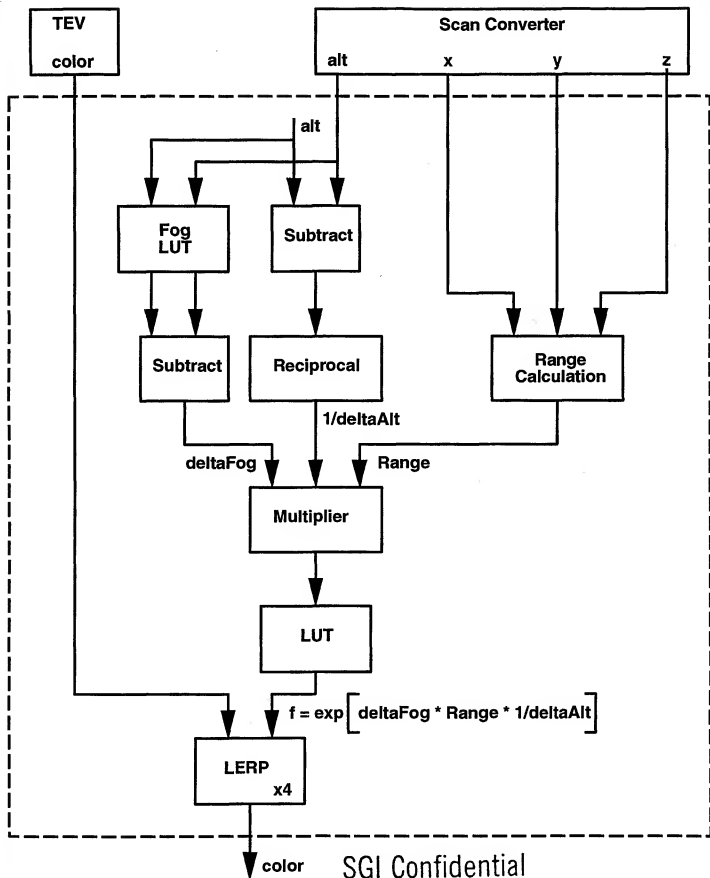
Ct: texture color
 At: texture alpha
 Cf: lit fragment color
 Cc: constant color
 Cb: bias color

Replace Fragment:
 Replace Texture:
 Modulate:
 Blend:
 Decal:
 Add

A = 1	B = 0	D = 0
A = 0	B = 1	D = 0
A = Ct	B = 0	D = 0
A = -Ct	B = Cc	D = Cf
A = -At	B = At	D = Cf
A = 1	B = Cc	D = Cb

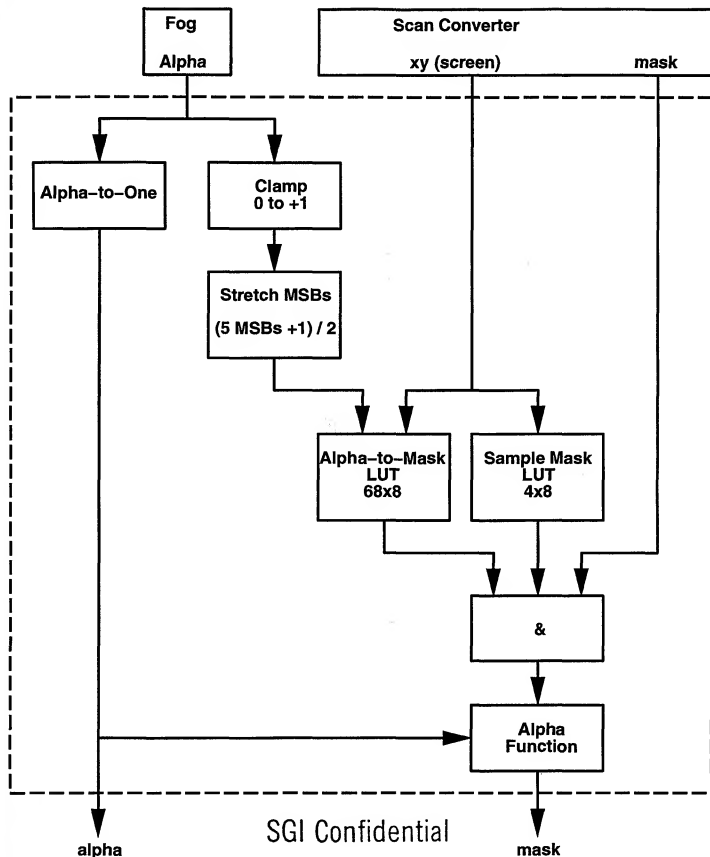
SGI Confidential

Fog Block Diagram



SGI Confidential

Alpha and Mask Block Diagram



SGI Confidential

FRU Functions

- * Stencil, depth, blend, logic-op non-multisampled pixels
- * Supported fb formats:
S10E5, RGBA12, LA16, CI16
- * Accumulation buffer
- * Fast clear
- * Pass fragment info to M chips, receive resolved color from M
- * Automatic & explicit Z cull
- * Calligraphics
- * In-place copy & blend
- * Partial resolve for multipass shading

28

SGI Confidential

32

FRU Performance

- * One depth/stencil tested, blended pixel per clock
- * One accum buf operation every 2 clocks
- * Fragment packet from ltf = 180 bits
- * Desired 1 pixel/clock performance = 143 MPixels/sec
- * Pixel = (RGBA, SZ) = $16 \times 4 + 32$ = 96 bits
- * RW of a pixel = 192 bits/clock = 3400MB/sec peak

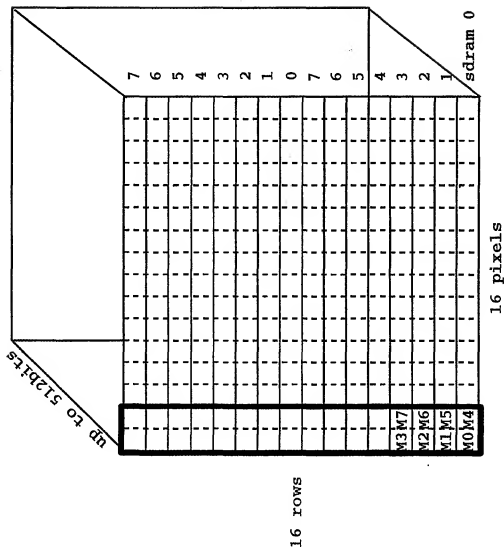
29

SGI Confidential

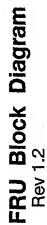
39

FRU sdram Memory Map

Rev 1 11/11/96



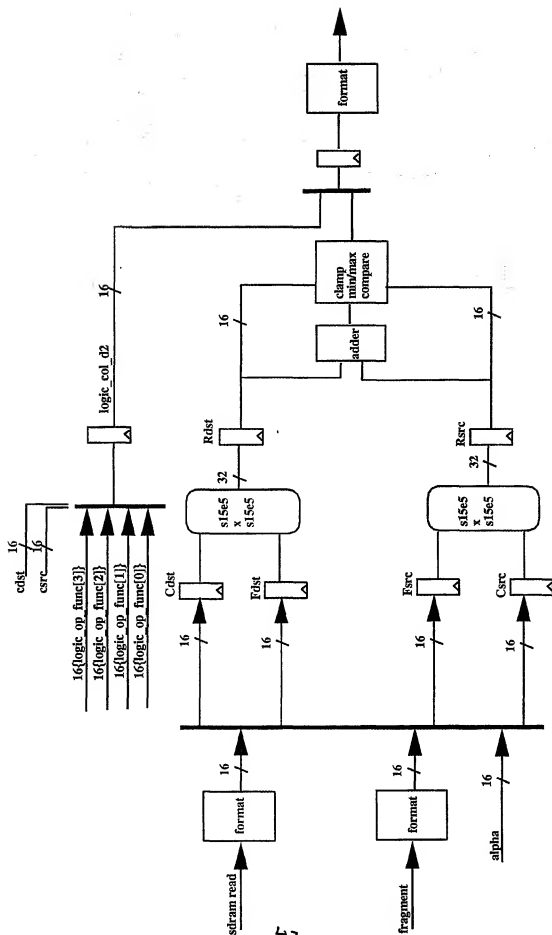
SGL Confidential



SGI Confidential

FRU Blend

Rev 1 11/11/96



SGL Confidential

42

FRU Issues

- * Tiling & load balancing**
- * Better solution for pixels-in-flight problem?**
- * FRU_TRB data**
- * Final supported pixel formats/conversion**
- * Additional blend functions?**
- * Alternative implementations**

SGL Confidential

R EDRAM Interface

R SDRAM & Network Interfaces

David Wang
John Montrym

http://b7.esd/ball/treen/doc/doc/chape/r/rinf_offsite2.no

Ball offense 2-

Vital Statistics

8 32-bit wide SDRAMs

$BW = 143MHz + 32bytes + 80\% = 3660 MB/sec$

- 360MB/sec in G FIFO write
- 360MB/sec out G FIFO read
- 70MB/sec out video refresh
- (1280x1024x2Hzx6bytes, 8R) texture response (limited by chip I/O)
- 105MB/sec out texture response (limited by chip I/O)
- 1816MB/sec in/out framebuffer
- 1.4GB/chips/s (bw.txt)

Memory size per R = 16MB (use 512Kx32 SDRAM)
32MB (use 2*1Mx16 SDRAM)
128MB (use 2*64Mx16 SDRAM)

Ball Offsets 2-

Design Goal

Command queue based interface allows fire-and-forget protocol

Dedicated queues for texture and framebuffer access

Separate queues for SDRAM bank A and B to hide page miss latency

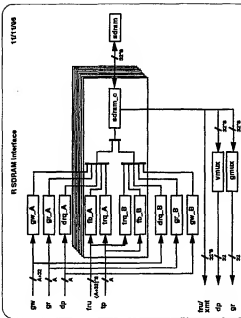
256-bit atom access allows zero overhead bank switch

Round-robin arbitration with flow control

display request has the highest priority

Display request can lock the current SPRAM to allow random access within the same page

Ball Office 20



R SDRAM Interface

Issues

Max display response latency

First size = 2 to allow issuing new page activate command in the middle of transfer

Bus bandwidth

- framebuffer A/32*8/32*8
- display processor A/32
- G FIFO read A/32
- G FIFO write A*32
- texture prefetch A/32*8

Support 4-bank SDRAM

Performance optimized arbitration

Ball Offsets 2

R Network Interface

Vital Statistics

Three 16 bit @ 143MHz*3 input and output channels

BW = 143MHz*3*2bytes*3 ports*75% = 1930MB/sec

R chip layout budget

- 160MB/sec in video requests
- 96MB/sec in video requests (roughly 1280x1024x2Hz 8 R system)
- leave 1518MB/sec for texture req & responses
- leave 1518MB/sec for texture data (responses)
- 1170 MB/sec for texture data (responses)

R chip output budget

- 48MB/sec out video responses
- 1280x1024x2Hz*8bytes, 8R
- 1561MB/sec out texture req & responses

Ball Offsets 2

R Network Interface

Design Goal - rcv

Match or provide higher BW than Ball net

Individual flow control for

- texture response packet
- display response packet
- G FIFO write packet

Throttle-back message for G-R flow control

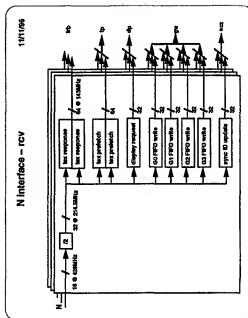
Round robin among channels

Need to re-sort packets from a specific G

Half tile responses per (R) cycle

- we can achieve ~0.7 tile per cycle

Ball Offsets 2

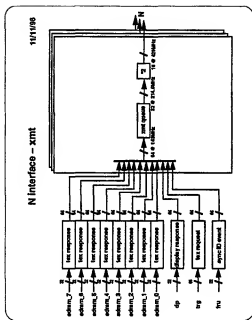


N Network Interface

Design Goal - xmt

Display response has the highest priority
Round robin arbitration for others
Half texture request per (R) cycle
- we can achieve one texture request per cycle

Ball Offset 2



N Network Interface

Gate Count

** adram **
% area: 2.7% of total chip core
Gates: 290.0K gates
ram: 0.0K bits
area: 12.11 sq-mm

** xmt **
% area: 4.1% of total chip core
Gates: 264.0K gates
ram: 24.0K bits
area: 21.11 sq-mm

** rwr **
% area: 2.5% of total chip core
Gates: 279.0K gates
ram: 0.0K bits
area: 11.02 sq-mm

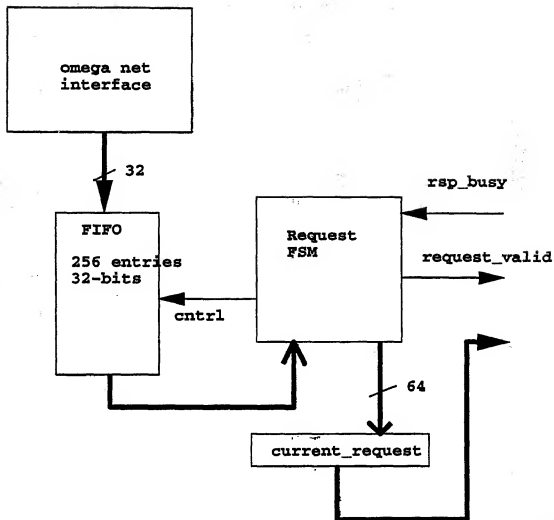
Ball Offset 2

N Network Interface

Issues

Three narrow N channels overhead
Throttle-back flow control scheme
Force a given G to use specific channel?
avoid sorting G packets
Error detection/correction
Need high speed (214.5MHz) SRAM for N write

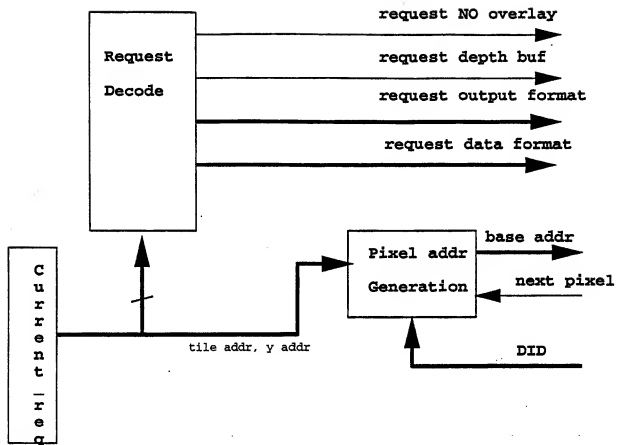
Ball Offset 2



RAM: 256 x 32 bits for FIFO
 FFLOPS: 64 current_request
 9 fifo rd pointer
 9 fifo wr pointer
 1 state bit for FSM
 1 request_valid
 1 rsp_busy

Clock count: 1 to get to front of FIFO,
 1 to get out of FIFO into Request FSM
 2 in the FSM.

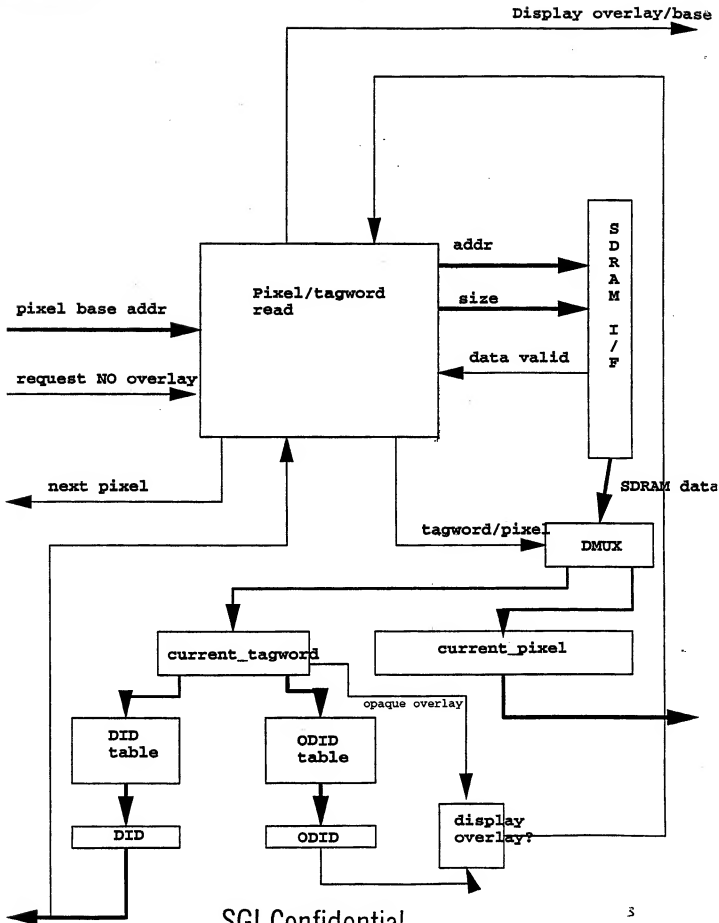
SGI Confidential

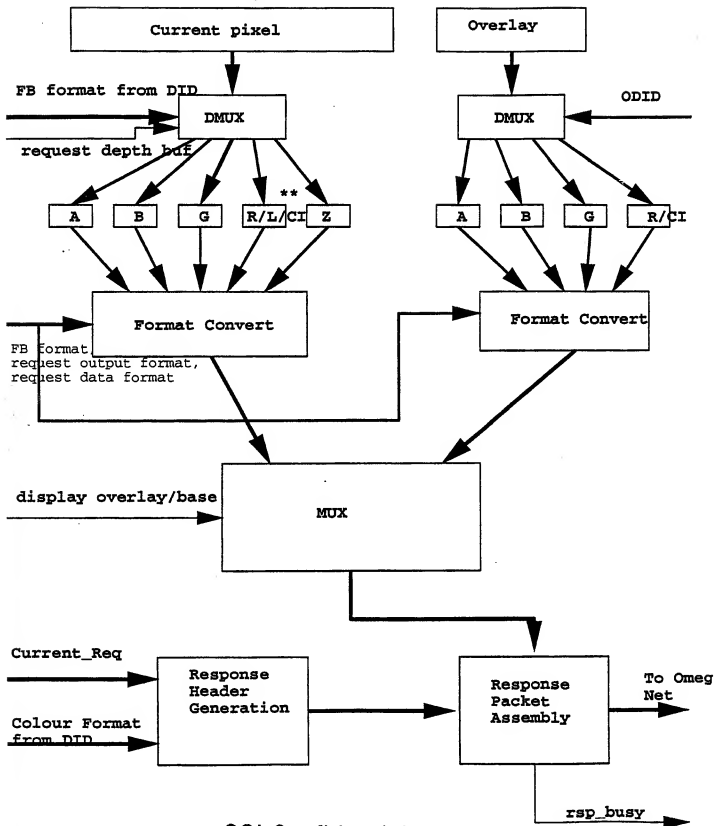


SGI Confidential

49

2





SGI Confidential

Overview

Scalar core programming environment
Pixel operation performance

Throughput

Setup

Open issues

Scalar Core

Pixel path

Scalar core

We are assuming:

200 MHz performance

RISC instruction set

Fast immediates & bitwise ops

Floating point support

Simulator w/ interactive debugger

Register writes

Immediate to G chip: 1 cycle

Immediate + mask to PE: 2 cycles

Fast interface to G & vector units

What if the core is external?

G chip register reads

Round-trips / semaphore checks

Low entry-point overhead

*→ aim used to have
unnecessary caller
state to stack.*

Pixel Peak: Simple stuff

(Assuming DUPLO interface) → *new design #s*

RGBAS DrawPixels

RGBAS ReadPixels

RGBAL2 copy

In-place

Not in-place

325 MPix/s

283 MPix/s

1000 MPix/s

536 MPix/s

Pixel Peak: Useful stuff

(Assuming DUPLO interface, 8 R's)

RGBAL2 copy

5x5 convolve

11x11 convolve

350 MPix/s

121 MPix/s

SGI Confidential

52

Pixel Setup: GI & GE

Write from host

IDMA Engine: ~20 cycles
Address, width, skip ~250 cycles
Debubler: ~80 cycles
Dead bytes for scan lines ~80 cycles
Converter: ~100 cycles
Command table: ~100 cycles
Total: **450 cycles**
Fetch unit: ~150 cycles
SDRAM addr, x, y, width, height ~80 cycles
Converter: ~80 cycles
Source type, dest type ???
ODMA Engine:

Pixel Setup: PE

Write from host:

Input/output types
SDRAM addr
x, y, width ~150 cycles
Reload from scalar: 0 cycles
Nothing?

Pixel Setup: R Chip to TFI

Promoter

~8 registers

Incoming format

~8 registers

TRG Texture filter mode

(convolution / color matrix)

TFI Filter mode

~4 registers

TFI Multiply/accum

121*4 coefficients (max) 242 words
4*4 coefficients (matrix) 8 word

Pixel Setup: LTF & IMP

LTF Pre-LUT scale/bias

8 registers

LTF LUT/histogram

Configure 8 registers
Load from SDRAM ???

LTF Post-LUT scale/bias

8 registers

8 PE shadow reg writes ~16 cycles

IMP Converter 4 registers

IMP Color Mask 3 registers

IMP Blending Mode 1 register

SGI Confidential

53

53

Pixel Setup: R Chip Total

Total: 51 registers
Excluding convolve coefficients
If written from scalar: ~100 cycles
Writes alone: ???
Get from EMEM: ???
Compute: ???
Should happen on mode change

Pixel Setup: Misc

DrawPixels overhead
Lazy mode stuff etc: ~180 cycles
Draw textured rectangle
Send directly to PE ~200 cycles
Restore G state 16 cycles
Restore PE state 16 cycles?
Polymode etc.
Restore R state
Is this necessary every time?
2 copies of shadow on PE?
Other lazy validation scheme?
Leave PE shadow in geometry state?
Pixel context switching expensive 4

Pixel Setup: Useful draws

DrawPixels (no transfer modes)
G chip setup: 700 cycles
R chip setup: 800 cycles
State restore: ??? cycles
Total: 1.5K cycles
Ops/sec: 133 K
DrawPixels (convolve)
G chip: 700 cycles
R chip (download): 800 cycles
R chip (draw): 800 cycles
Draw rectangle: 200 cycles
State restore: ??? cycles
Total: 2.5K cycles
Ops/sec: 80 K

Pixel Setup: Useful copies

CopyPixels w/ LUT & scale/bias
Copy to scratch: 200 cycles
Configure R chip: 200 cycles
Draw rectangle: 200 cycles
Total: 1.2K cycles
Ops/sec: 166 K
CopyPixels w/ small convolution:
Copy to scratch: 200 cycles
Configure R chip: 1000 cycles
Draw rectangle: 200 cycles
Total: 1.4K cycles
Ops/sec: 140 K
CopyPixels w/ LUT & convolve
Copy to scratch: 200 cycles
Configure R chip: 1800 cycles
Draw two rectangles: 400 cycles
Total: 2.2K cycles
Ops/sec: 90 K

SGL Confidential

Biggest Unresolved Issues

Scalar core

Are our expectations realistic?

What if it's external?

Code space / cache requirements

Pixel Path

More specifics on setup

Software design / code cuts

R chip control / code cuts

HW state management

Mode change speeds

What can we simplify?

Pixel textures

Multiply/accumulate

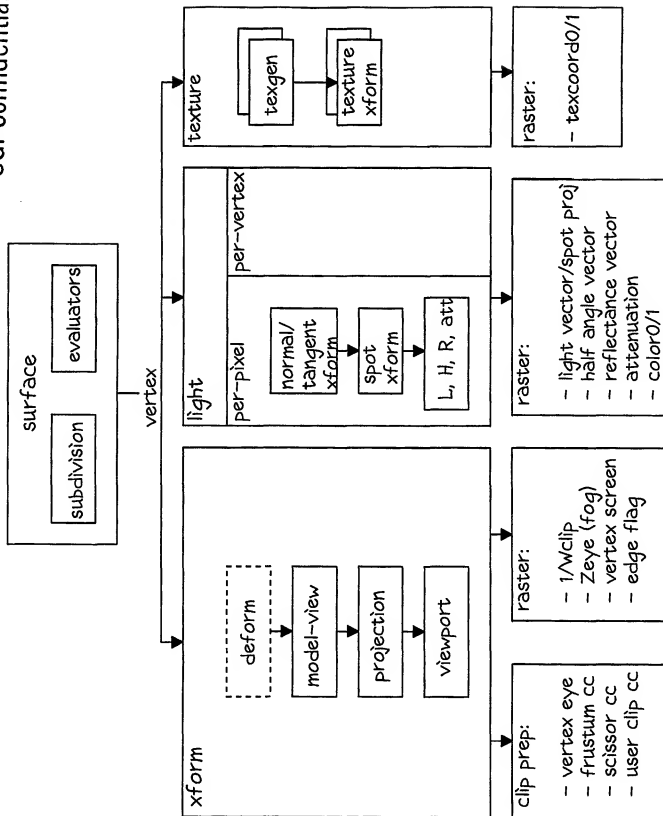
Multiple spigots

SS

SGI Confidential
SS

Geometry Path

SGI Confidential



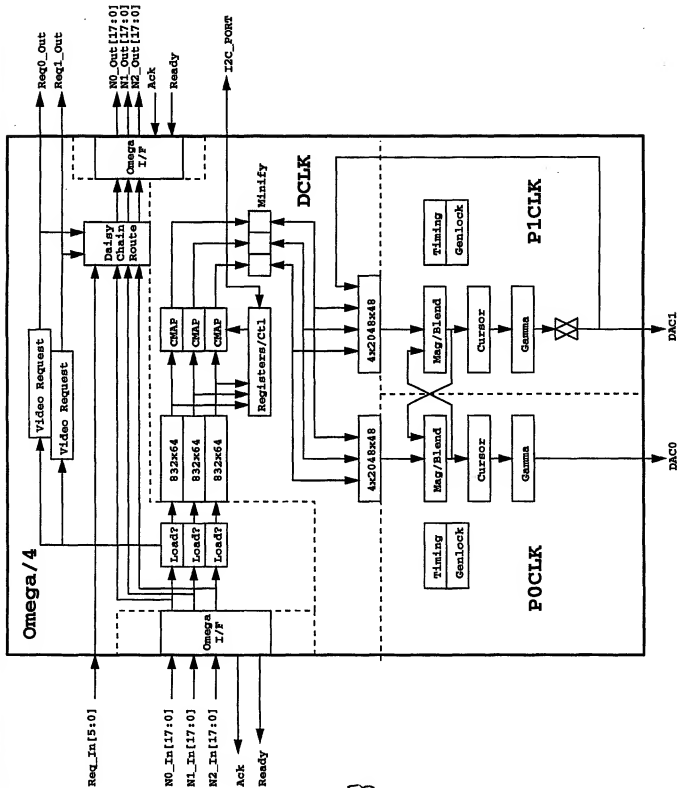
Geometry Path (cnt'd)

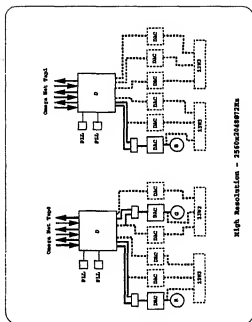
- o What's new:
 - Better surface to vertex integration
 - Per-pixel lighting support
 - No overhead primitive switch
 - Parallel matrix operations
 - Double buffered zform states
 - Multiple projection and viewport definitions

Software Development Environment

- o G software simulator for code development
- o Vector code scheduler, assembler (back end)
- o Scalar and vector code segmented, shared libraries

SGL Confidential





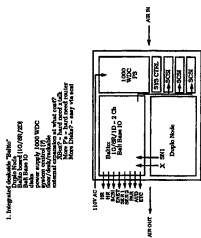
SGI Confidential
61

Flashlight 80K6

"Via Console"
 Either GM10 in Lego (now)
 or
 GM20 in Lego (eventually)
 or
 GM20 in Lego (now)

Hile Base IO (MIO)
internal acd
external acd
4 serial
2 ihd
2 mss
parallel
headphones
microphone
audio stereo in/out
cnet
ADAT in/out
genlock in/out
speaker power
digital in/out
interrupt in/out

Lessons Learned



III

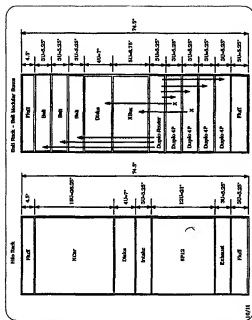
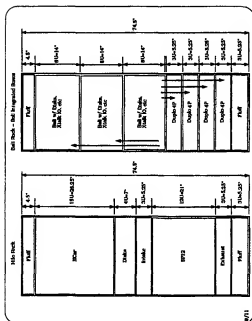
[illegible]

Area 11 New Q3

Ball Extended Base I/O
internal serial
external serial
4 serial
2 kbd
2 mae
parallel
headphones
microphone
audio stereo in/out
enet
ADAT in/out
genlock in/out
speaker power
digital in/out
tape in/out

Lima 31 Oct 96





Ball Major Product Design Tradeoffs

1. Integrated = Lowest Cost
2. Modular = Highest Flexibility
3. Define the configurations needed and packaging will follow.
4. Risk optimization limits size, shape, airflow, etc.)
5. Yage and Kier represent current standards for I/O, density, etc.

Ball Major Product Design Questions

1. Board/chip module level interconnect feasibility and size, service, assembly, cost
2. Power/Cooling requirements of worst case Ball
3. Power supply leverage/new development
4. Ball - Integrated or Modular - optimize for what config?
5. Duplo I/O for desktop - Integrated Xtalk I/O and clocks or Xbus and Databus? Assume we need Xtalk bandwidth for video compression etc.

SGI Confidential

The M Chip

Mark Leather

Shore Graphics - Confidential

What is the M chip?

- The M is a hybrid ASIC containing ~100K gates of standard cell area, and 10 Mbits of DRAM arranged in four banks of 2.5 Mbits each
- It's only purpose is to perform multisampling. It contains all multisample memory on-chip allowing very high bandwidth access.
- The M chip accepts fragments from the R, renders them into multisample memory and returns a rescaled color for each fragment.

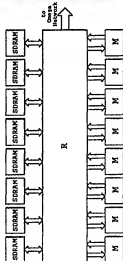
Shore Graphics - Confidential

Why do we need an M chip?

- The idea for the M chip arose as a possible solution to the DRAM bandwidth bottleneck problem (while multisampling) - a factor of 6 to 8X bandwidth was required over what was available.
- Two other solutions were proposed:
1. Sort incoming primitives into screen lines, and rely on caching to reduce DRAM bandwidth.
2. Find a new anti-aliasing algorithm which is less bandwidth intensive.
- Solution 1 was proposed by Doug Vorhies. It seemed promising at first, but was later abandoned due to difficulties in implementing the full OpenGL feature set.
- Solution 2 was investigated by several people including Alex Stehly, Bob Drach, and Mark Leather. The best solution offered only a 3-4x average saving in bandwidth.

Shore Graphics - Confidential

R Subsystem Block Diagram



Shore Graphics - Confidential

SGI Confidential

Performance:	17.375 Million Pixels/second
Pixel Size:	8 samples, 64 bits/sample (medium format) 8 samples, 96 bits/sample (large format)
Pixel Capacity:	10 Mbits = 20 K Pixels (medium format) 13 K Pixels (large format)

Internal clock: 143 MHz

to clock; 386 MHz or 143 MHz

Booked on 07/25/2008

[illegible]

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
84

Abstract

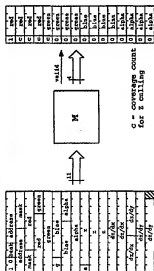
000026 1 : 14001 0001/2 20 14001 0001/2 14001 0001/2

AP
GUYTON

17005 2003 03 715

Silicon Graphics - Confidential

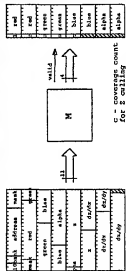
Input/Output Packet Format (286 MHz)



note: There is no handshake. Output data is generated exactly n clocks after the input is received. The '1' in the output data is required to denote the start of the packet because the data is source synchronous.

Public Commentation – Confidential

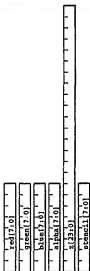
Input/Output Packet Format (143 MHz)



note: There is no handshake. Output data is generated exactly a clock after the input is received. The '1' in the output data is required to denote the start of the packet because the data is source asynchronous

Evening Questions – Confidential

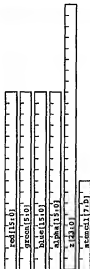
Medium Sample Format



total, 64 bits

Abstract: Quantitative - Content Analysis

Large Sample Format

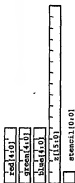


Total: 96 bits

notes: An additional bit may be required for z culling. This can be taken from either s or extend.

SGI Graphics - Confidential

Small Sample Format



Total: 64 bits

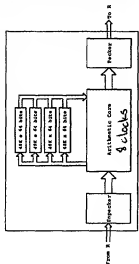
note: This format is not in the original proposal. It is a reduced cost system with 4 M's per S.

r, g and b are dithered based on sample position to give an effective resolution of 8 bits per component on the sample position.

z is compressed using the Runa 16 bit compression method

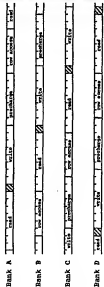
SGI Graphics - Confidential

M Chip Block Diagram



SGI Graphics - Confidential

M Chip DRAM Timing



note: Input addresses must be received in sequential order to start banks A, B, C, D, A, B, C, D etc.

SGI Graphics - Confidential

SGI Confidential

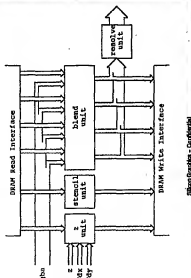
Distribution of M's and DRAM banks within an 8x8 screen tile

7D	6C	9B	4E	7B	3C	1B	0A
2B	1F	0C	7A	6A	5D	4C	
5C	4D	7A	6D	1C	0D	7B	2B
1A	0B	3C	2D	5A	4D	7C	6A
6B	7C	4F	5B	2D	3C	0B	1A
2B	3A	0D	1C	6B	7A	4D	5C
4C	5D	6A	7B	8C	1D	2B	3D
0A	1F	2C	3D	4A	5B	6C	7D

note: This is one of several possible tiling's.

Show Diagram - Confidential

Block Diagram of Arithmetic Core



Show Diagram - Confidential

Possible Configurations

8R (64M) system:	1024x1024	8 large samples
	1280x1024	8 large samples
	1280x1024	4 large samples
	1664x1216	4 small samples
	1920x1088	4 small samples
16R (128M) system:	1280x1024	8 large samples
	1664x1216	8 small samples
	1664x1216	4 large samples
	1920x1088	4 small samples
	1920x1088	4 large samples

Show Diagram - Confidential

small = med from prev slides.

Open Issues

- Selection of vendor. This may affect the on-chip DRAM configuration, which could affect the architecture. As an example, one chip manufacturer can supply a single, 2-bank, 256-words DRAM running at 100MHz. Although this meets our bandwidth requirements, it is unlikely we could use this without a radical redesign of the chip's internal architecture.
- Additional functionality:
 - Order independent transparency (Fragment scheme or dual Z buffer)
 - CSG
 - Fast Multisample Clear and Copy
 - Funny read (just started to normalize each resolved color)
 - Color correction
- Support for systems with less than 8 M's per R
- Support for 16 samples per pixel at reduced speed

Show Diagram - Confidential